

Research on DB2 for z/OS Application Performance Tuning

Zhuang Huanhuan¹, Gao Zhen¹, Yang Jian², Wang Min¹

¹School of Software Engineering, Tongji University, Shanghai, China

²Software School, Fudan University, Shanghai, China

¹zhuanghuan1986@gmail.com, gaozhen@tongji.edu.cn, ²yangjian@fudan.edu.cn

Abstract -DB2 for z/OS is a successful commercial DBMS which is widely used worldwide especially in global top banks, retailers and health insurance providers etc. The paper focuses on the research of DB2 for z/OS application performance tuning, and purpose of the paper is to provide a systemic method that can tune DB2 for z/OS to get better performance. The systemic method includes five different parts: Tablespace tuning, Bufferpool tuning, Lock size tuning, Isolation level tuning and Access path tuning by using MQT. Concrete lab materials and results are provided to verify the proposed methods.

Keywords-Performance Tuning; DB2 for z/OS; Table Space, MQT

I. INTRODUCTION

In multitudinous commercial Relational database, DB2 for z/OS in the mainframe platform of IBM is considered as one DBMS system with best performance. The top 56 banks in the world, 23 of the best 25 retailers in America, 9 top healthcare centers around the world are using this commercial database. We put forward higher requirements on performance of DB2 for z/OS for the reason that there are massive data stored. The research on optimizing DB2 for z/OS application system becomes more and more urgent and important.

Application tuning of database contains each part of system's design, implementation, operation and maintenance, this paper focuses on the research of how to tune DB2 for z/OS in its design and implementation phase.

II. THE METHODS ON PERFORMANCE TUNING OF DB2 FOR z/OS

The main work for tuning DB2 for z/OS is database tuning to increase the throughput of database, and shorten events' Response time. In order to do database tuning, we should firstly find out the bottle-neck that affects the database performance. Secondly, set a reasonable goal for performance. Then, we find out the factors which affect the database performance. At last, we carry out specific tuning works according to the Factors mentioned above. There are three common methods for tuning DB2 for z/OS: (1) Tuning of storage structure; (2) Tuning of concurrent control; (3) Defragmentation of DB2; (4) Tuning of SQL access path. In this paper we use a minibank system as a case to show the tuning methods of DB2 for z/OS' application system.

III. CASES STUDY OF PERFORMANCE TUNING OF DB2 FOR z/OS

This section will focus on the tuning methods such as table space, buffer pool, lock granularity, isolation level and Access

path by using MQT, also we will test and analyze the system before and after tuning.

A. Tuning of Table Space Design

The logical data in DB2 for z/OS application system should be stored in physical disk as a table space. We should use different table space as the scale of data differs, otherwise, some performance problems will be caused. Suppose a table with 500 thousands records in it, we used Segment table space and Partitioned table space to store the data respectively, then do random update test. Total time consumption and time distribution in two projects are shown in the Figure 1 and Figure 2. Total time consumption of Segment table space is 17.39.1163 minutes, 11.46.7316 minutes for Partitioned table space.

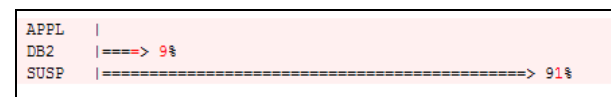


Figure 1. Time distribution of segment table space

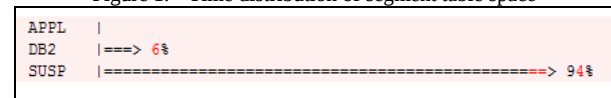


Figure 2. Time distribution of partitioned table space

Furthermore, we classify the time consume of two projects, found that the CPU consume time was almost the same in two projects. However, the difference on I/O waiting time is obvious. The I/O in Partitioned table space was 6 minutes faster than Segment table space in a data-operation-intensive system. It is shown as follow in the Table 1 and Table 2.

TABLE I. THE TIME CONSUMED BY CPU AND BY WAITING

	Segmented Table Space	Partitioned Table Space
DB2 CPU time (m)	1:08.29719	1:03.08426
Waiting time (m)	16:32.471254	10:42.458093

TABLE II. THE TIME CONSUMED BY DATABASE I/O AND BY OTHERS

	Segmented Table Space	Partitioned Table Space
DB2 I/O time(m)	16:19.364913	10:24.244841
Other time (m)	<14s	<13s

By analyzing the data in table, we would see that when frequently random operation executed on a large scale data table, Partitioned table space design performs better than Segment table space.

B. Tuning of Buffer Pool Design

Buffer Pool is an area in memory that is used to store the user's table thread by DB2 for z/OS. Buffer Pool is divided into many units which are called pages. The size of one page could be designed for 4K, 8K, 16K or 32K.

To figure out how the size of buffer pool can affect database querying performance, we designed a test case like this: importing 100,000 threads into the same transaction records table with the Load Utility in DB2 for z/OS, by using a table space with 4K page buffer pool and 32K page buffer pool respectively to compare the time consuming. The test result showed that the time consuming of 4K buffer pool design was 32.598614s, and the time consuming of 32K buffer pool design was 2.616713s, which is greatly reduced. It is shown in Figure 3 and Figure 4.

ELAPSED TIME DISTRIBUTION	
APPL	=====> 16%
DB2	=====> 23%
SUSP	=====> 60%

Figure 3. Time distribution of 4K buffer pool design

ELAPSED TIME DISTRIBUTION	
APPL	=====> 16%
DB2	=====> 83%
SUSP	=> 2%

Figure 4. Time distribution of 32K buffer pool design

By analyzing the distribution of time consuming, we obtain TABLE III, from which we could see that the waiting time is the key factor that causing performance difference. The I/O time of 32K buffer pool was almost only 0s, but it was 29s if buffer pool size was 4K. On the other hand, the hit rate of 4K buffer pool was 3.83%, 51.46% in 32K buffer pool. The average length of thread in current transaction record table was 2257B, 32K page could store more threads than 4K page, it performs better on disk I/O operation.

TABLE III. THE DETAILED TIME DISTRIBUTION OF BUFFER POOL TEST

Design patterns	Design of 4K buffer pool	Design of 32K buffer pool
Total time-consuming(s)	32.598614	2.616713
DB2 elapsed time (s)	32.193364	2.210271
Waiting time (s)	29.451745	0.043572
Other I/O time (s)	29.443876	0
the number of pages received from the BP4K	100,021	3
the number of pages received from the BP32K	790	8649
Buffer pool hit ratio	3.83%	51.46%

This test case tells us that we should choose the size of buffer pool according to the length of thread in table. The longer thread should be stored in bigger pages, for the purpose of increasing utilization rate, and then increase the hit rate of buffer pool, thereby increasing the efficiency of program.

C. Tuning of Lock Size Design

For large-scale online transactions, it would probably cause a long time waiting, even a dead lock if the lock size design was not reasonable. In the following test case, we compare the performances of a drawing money transaction by

changing table space lock size, using page lock and row lock respectively. We test like this: running the same number transactions in design proposals using different lock size, then checkout the average time consuming of all transaction. According to test result, under page lock situation, 10000 transactions need 2m23s39, when using row lock, the time consumption is 2m15s92. The contrast of transaction response time in two lock size solutions is shown in Figure 5..

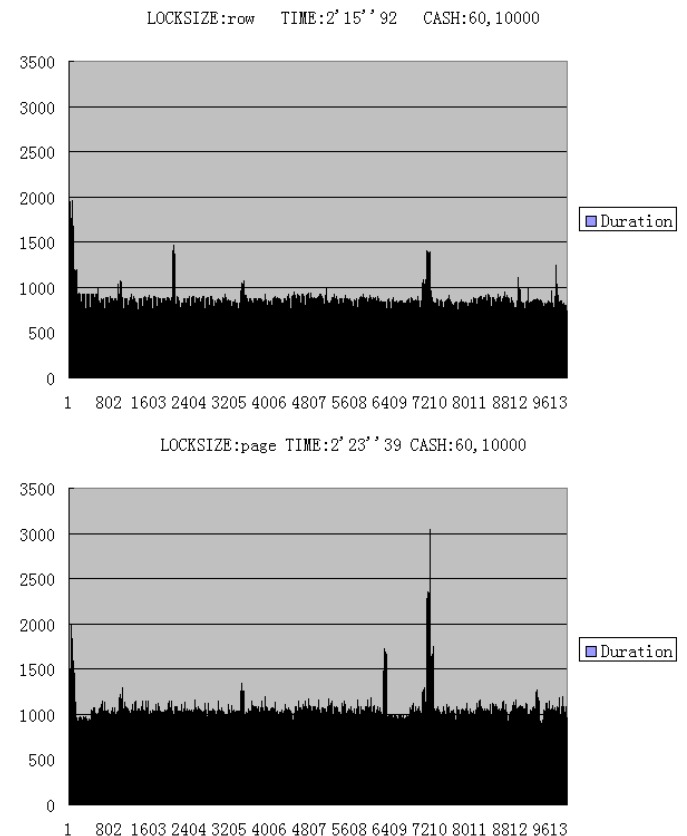


Figure 5. The contrast of transaction response time in two lock size solutions

In page lock solution, the average time consuming is 0.187573s; in row lock solution, the average time consuming is 0.184944s. The CPU time of DB2 and Suspend time are shown in TABLE IV, the contribution of suspend time is shown in TABLE V.

TABLE IV. THE TIME DISTRIBUTION OF DB2 AND SUSPEND TIME

	Page-level lock granularity	Row-level lock granularity
DB2 CPU time (s)	0.006993	0.006925
Suspend time (s)	0.155470	0.147587

TABLE V. THE TIME CONSTITUTION OF SUSPEND

	Page-level lock granularity	Row-level lock granularity
Lock waiting time (s)	0.134500s	0.123592
Other Time (s)	<0.021s	<0.021s

From these tables, we can see that compared to page lock size, row lock size solution using a shorter and more stable transaction time, a better performance, and a fast average time. The time consuming of DB2 I/O is almost focus on Suspending, in which lock waiting time occupied the most of total time. The waiting time of page lock is longer than the row lock.

By comparative analysis, we found it is the difference between lock size causing the difference of lock waiting time, and then determine the difference of online transaction performance. To sum up, we should ensure the smallest lock size when the system resources is not limited, for avoiding the competition of resources, and decreasing the waiting time, then improving the system concurrency.

D. Tuning of Isolation Level Design

Under situation of large-scale transactions, the isolation level will directly affect the performance of the transaction response time and performance. In the following chapter we test and compare two isolation levels: the stability of RS and RR Repeatable Read isolation level. The test method is as follow: run the same number of online transactions in design proposals using different isolation levels, and then calculate the average time-consuming per transaction. When using RS isolation level, 4944 transactions are responded; and 4922 transactions are responded using RR isolation level. The trend of running time per transaction is shown in Figure 6 and Figure 7. In RS isolation level, the average transaction time-consuming is 0.683128s; In RR isolation level, The average transaction time-consuming is 0.732747s.

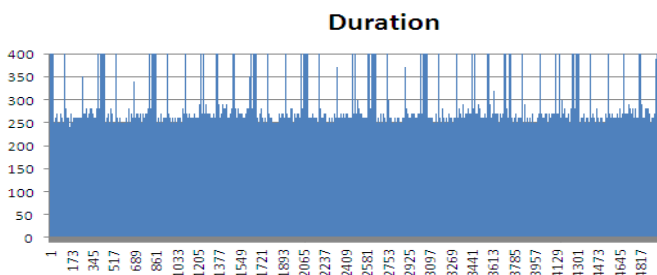


Figure 6. Transaction response time distribution under RS Isolation Level

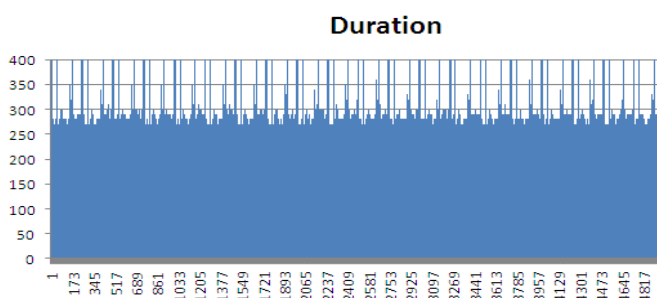


Figure 7. Transaction response time distribution under RR Isolation Level

We can observe the time composition of transaction using DB2 Trace, A great difference of internal time consuming is found, especially the waiting time, both the two isolation levels spending lots of time on lock waiting, but the RR's waiting time is four times as RS's

RS isolation level only lock all rows that are involved in the current program, but the RR isolation level will lock the entire table where these rows in, so the latter doesn't perform as well as the former. To choose an isolation level, we should not only consider for the security of data, or the response time, we should make a choice according to the specific business. We should find a balance of them, which is the essence of isolation level optimization.

E. Tuning of Query Access Path Design

MQT (materialized query tables) can significantly improve query performance, especially for complex queries. In this chapter we do comparison test by closing and opening MQT mechanism. Run the same queries in two test program, check out the response time and the access path respectively. The total time-consuming was 4.140773s when MQT is used, otherwise the time-consuming is 0.042679s. Specific time distribution is shown in Figure 8 and Figure 9.



Figure 8. Time distribution without MQT open

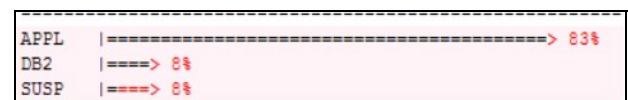


Figure 9. Time distribution with MQT open

Test data show that the query response time is optimized by geometric times when using MQT, access paths were simplified several times compared to not using MQT. The total time-consuming and the distribution of all parts are shown in TABLE VI..

TABLE VI. TIME DISTRIBUTION OF MQT TEST

	MQT is not open	Open MQT
Total time (s)	4.140773	0.042679
DB2 elapsed time (s)	3.538491	0.007004
CLASS1 CP CPU	6846	23
Other I / O time (s)	0.047307	0
4K BP pick pages	7321	4

MQT read and store all the underlying table first, when the data is accessed by some operation, MQT can offer the data in a quick and direct way, which greatly improve the performance of inquiry. But we should be very careful when using MQT, it's a good choice to use MQT when the underlying tables don't change frequently, for the reason that MQT doesn't update data when the underlying table changes.

IV. CONCLUSION

This paper researches performance optimization technology of commercial database DB2 for z/OS. We use several methods on table space tuning such as Buffer pool tuning, Lock size tuning, Isolation level tuning and Access path tuning by using MQT. By using a small bank application system as an instance, we test these methods. In this way we received detail experiment data, which is shown in a vivid way by charts and tables.

ACKNOWLEDGEMENT

This paper was funded by IBM CDL(China Development Laboratory).

REFERENCES

- [1] B. Schiefer and G. Valentin. DB2 Universal Database Performance Tuning, IEEE Data Engineering Bulletin 22(2), June 1999, pp. 12 - 19.
- [2] G. Weikum, A. Christian, A. Kraiss and M. Sinnwell. Towards Self-Tuning Memory Management for Data Servers, IEEE Data Engineering Bulletin 22(2), June 1999, pp. 3 - 11.
- [3] DB2 Version 9.1 for z/OS Administration Guide (SC18-9840-03),IBM Corp., Third edition,February 2008.
- [4] DB2 Version 9.1 for z/OS SQL Reference (SC18-9854-05). IBM Corp., Fourth edition, February 2008.
- [5] DB2 Version 9.1 for z/OS Utility Guide and Reference (SC18-9855-03). IBM Corp., Third edition, February 2008.
- [6] Craig S.Mullins. Evolve IBM DB2 UDB SQL access path. (In Chinese) <http://www.ibm.com/developerworks/cn/data/library/techarticles/0301mullins/0301mullins.html>
- [7] Wiese, David; Rabinovitch, Gennadi, “Knowledge Management in Autonomic Database Performance Tuning” , 20-25 April 2009.
- [8] S.F.Rodd, Dr. U.P.Kulkarni, “Adaptive Tuning Algorithm for Performance tuning of Database Management System”, (IJCSIS) International Journal of Computer Science and Information Security, Vol. 8, No. 1, April 2010.